2023 年 6 月 15 日（木）実施

反復構造のプログラム

1)　0 回以上の繰り返しのプログラム

for 文

C#言語で 0 回以上の繰り返しのプログラムを実現するための文として，for 文と while 文とが用意されている。for 文の構文は次の様になる。

```
for（ 初期化処理; 継続条件式; 更新処理 ）{ 文 }
```

先ず，初期化処理を実行し，続いて継続条件式を評価する。ここで，継続条件式が true であれば，文を実行する。次に更新処理を実行した上で，再度，継続条件式を評価する，という繰り返しを行う。継続条件式が true でなければ，文を実行せず，for 文から抜け出す。最初から継続条件式が true でなければ，全く文を実行しないので，0 回以上の繰り返しと呼ばれる。

for 文は通常，**特定の処理を決まった回数繰り返す目的**で用いられる。

＊ for 文の括弧の中の 3 つのセクションには，それぞれ，順に initializer，condition，iterator という呼び名が付けられている。

while 文

while 文の構文は次の様になる。

```
while（ 継続条件式 ）{ 文 }
```

先ず，継続条件式を評価し，true であれば文を実行し，再度，継続条件式を評価する，という繰り返しを行う。継続条件式が true でなければ，文を実行せず，while 文から抜け出す。最初から継続条件式が true でなければ，全く文を実行しないので，0 回以上の繰り返しと呼ばれる。

while 文は通常，**処理を特定の状態になるまで繰り返す目的**で用いられる。

2)　1 回以上の繰り返しのプログラム

do 文

C#言語で 1 回以上の繰り返しのプログラムを実現するための文として，do 文が用意されている。do 文の構文は次の様になる。

```
do { 文 } while（ 継続条件式 ）;
```

先ず文を実行し，続いて継続条件式を評価し，true であれば再度文を実行し，継続条件式を評価する，という繰り返しを行う。継続条件式が true でなくなれば，文を実行せず，do 文から抜け出す。最初から継続条件式が true でなくとも，先ず文を実行するので，1 回以上の繰り返しと呼ばれる。

do 文は通常，**ある処理を実施し，その処理を特定の状態になるまで繰り返す目的**で用いられる。

3) 1 次元配列の扱い

foreach 文

C#言語で 1 次元配列を扱う場合，for 文で配列の添え字を利用して全要素に働き掛けることは可能であるが，foreach 文を用いると，全要素に対する処理を簡潔に記述出来る。foreach 文の構文は次の様になる。

```
foreach (データ型 変数名 in 配列名) { 文 }
```

（使用例）

```
int[] num = { 1, 2, 3, 4, 5};
int sum = 0;
foreach (int n in num)
{
    sum += n;
}
```

（等価な for 文）

```
for (int i = 0; i < num.Length; i++)
{
    sum += num[i];
}
```

配列の要素数

＊ 多次元配列では，入れ子になった for 文を使用した方が，配列要素を処理する順序をより厳密に制御出来る。

本日の課題

今回の授業では，レジスタアプリの作成を通じて，for 文及び foreach 文を取り扱う方法を学ぶ。

手順

1）コーディング

コードエディタを開き，クラスのフィールドで変数，定数及び配列の新規の宣言及び書き換えを行う。ここで，const 修飾子は定数を表す。

```
15      public partial class Form1 : Form
16      {
17          int count = 0;
18          int total = 0;
19          int taxsum = 0;
20          string receipt = "レシート¥n";
21          int countud = 0;
22          int b33flag = 0;
23          int pretotal = 0;
24          int pretaxsum = 0;
25          string sprod;
26          const int MAXNUM = 10;
27          const int MAXPROD = 12;
28          int[] prod = new int[MAXNUM];
29          int[] num = new int[MAXNUM];
30          int[] upc = new int[MAXNUM];
31          int[] downc = new int[MAXNUM];
32          int[] subtotal = new int[MAXNUM];
33          int[] subtax = new int[MAXNUM];
34          string[] subrec = new string[MAXNUM];
35          int[] pflag = new int[MAXPROD];
36
37          readonly int[] price = new int[] { 450, 380, 1000,
38              520, 350, 390, 290, 150, 198, 250, 230, 210 };
```

書き換え

それぞれの変数の意味と名称との対応は，アップダウンボタンの位置を表すカウンタ countud，button33用のフラグ b33flag，直前の合計 pretotal，直前の税額合計 pretaxsum，商品名 sprod である。定数の意味と名称との対応は，商品登録件数の最大数 MAXNUM，商品種別の最大数 MAXPROD である。また，配列の意味と名称との対応は，商品番号 prod，アップボタン押下数 upc，ダウンボタン押下数 downc，1件の金額小計 subtotal，1件の税額小計 subtax，レシートの内訳 subrec，商品ボタン用のフラグ pflag である。

次に，taxIncludedメソッドの定義を変更する。変更点は，第3引数として何件目の商品登録かを受け取ることと，税額としては小計を求めることの2点である。

```
81      private int taxIncluded(int p, int t, int i)
82      {
83          int ti;
84
85          ti = (int)(p * (1 + (double)t / 100));
86          subtax[i] = ti - p;
87
88          return ti;
89      }
```

ここで，LabelButtonInvisible メソッドを次の様に新規に作成する。

```
106     private void LabelButtonInvisible()
107     {
108         for (int i = 0; i < MAXPROD; i++)
109         {
110             pflag[i] = 0;
111         }
112
113         foreach (string s in lproduct)
114         {
115             labelSet(s, "");
116         }
117
118         foreach (string s in lnum)
119         {
120             labelSet(s, "");
121         }
122
123         foreach (string s in lprice)
124         {
125             labelSet(s, "");
126         }
127
128         foreach (string s in upb)
129         {
130             buttonVis(s, 0);
131         }
132
133         foreach (string s in downb)
134         {
135             buttonVis(s, 0);
136         }
137     }
```

続けて，イベントハンドラ Form1_Load の処理内容の記述を次の様に変更する。

```
91      private void Form1_Load(object sender, EventArgs e)
92      {
93          for (int i = 0; i < MAXNUM; i++)
94          {
95              num[i] = 1;
96              upc[i] = 0;
97              downc[i] = 0;
98              subtotal[i] = 0;
99              subtax[i] = 0;
100             subrec[i] = "";
101         }
102
103         LabelButtonInvisible();
104         labelSet("label31", "");
105     }
```

ここで，labelDisp メソッド及び setValues メソッドを新規に作成する。

labelDisp メソッドは第 1 引数で商品番号を受け取り，第 2 引数で何件目の商品登録かを受け取る。

```
154     private void labelDisp(int i, int c)
155     {
156         labelSet(lnum[c], num[c].ToString());
157         labelSet(lprice[c], "×" + String.Format("{0:#,0} 円", price[i]));
158         subtotal[c] = taxIncluded(price[i] * num[c], tax[i], c);
159         subrec[c] = sprod + " " + num[c].ToString() + "個×"
160             + String.Format("{0:#,0} 円¥n", price[i]);
161     }
```

また，setValues メソッドは引数で何件目の商品登録かを受け取る。

```
163     private void setValues(int c)
164     {
165         total = pretotal + subtotal[c];
166         taxsum = pretaxsum + subtax[c];
167         labelSet("label31", "合計 " + String.Format("{0:#,0} 円", total));
168
169     }
```

イベントハンドラ button21_Click の処理内容の記述を，labelDisp メソッド及び setValues メソッドの呼び出しを用いて，次の様に変更する。

（図は次のページ）

```
136         private void button21_Click(object sender, EventArgs e)
137         {
138             if (pflag[0] == 0)
139             {
140                 sprod = button21.Text;
141                 labelSet(lproduct[count], sprod);
142                 buttonVis(upb[count], 1);
143                 buttonVis(downb[count], 1);
144                 pretotal = total;
145                 pretaxsum = taxsum;
146                 prod[count] = 0;
147                 labelDisp(0, count);
148                 setValues(count);
149                 count++;
150                 pflag[0] = 1;
151             }
152         }
```

フォームデザイナー上で button22 をダブルクリックして，イベントハンドラ button22_Click の処理内容（赤枠の部分）を記述する。

```
171         private void button22_Click(object sender, EventArgs e)
172         {
173             if (pflag[1] == 0)
174             {
175                 sprod = button22.Text;
176                 labelSet(lproduct[count], sprod);
177                 buttonVis(upb[count], 1);
178                 buttonVis(downb[count], 1);
179                 pretotal = total;
180                 pretaxsum = taxsum;
181                 prod[count] = 1;
182                 labelDisp(1, count);
183                 setValues(count);
184                 count++;
185                 pflag[1] = 1;
186             }
187         }
```

イベントハンドラ button33_Click の処理内容の記述を次の様に変更する。

```
189         private void button33_Click(object sender, EventArgs e)
190         {
191             if (b33flag != 0)
192             {
193                 receipt = "レシート¥n";
194             }
195
196             for (int i = 0; i < MAXNUM; i++)
197             {
198                 receipt += subrec[i];
199             }
200
201             MessageBox.Show(receipt + "合計 " + String.Format("{0:#,0} 円¥n", total)
202                 + " (税額 " + String.Format("{0:#,0} 円", taxsum) + ") ");
203
204             b33flag = 1;
205         }
```

更に，アップボタン，ダウンボタンのイベントハンドラを作成する。先ず，フォームデザイナー上で button1 及び button2 をダブルクリックして，イベントハンドラ button1_Click 及び button2_Click の処理内容を次の様に記述する。

ここで，if 文による処理は，アップボタンが表示されたばかりの状態では，商品ボタン側で登録商品が何件目かを表すカウンタ count に 1 を加えているので，何件目のラベルに表示するかを表す変数 countud の値として，count から 1 を引いた数を代入している。

```
        1 個の参照
207     private void button1_Click(object sender, EventArgs e)
208     {
209         if (upc[0] == 0)
210         {
211             countud = count - 1;
212         }
213
214         num[0]++;
215         labelDisp(prod[countud], countud);
216         setValues(countud);
217         upc[0]++;
218     }
```

```
        1 個の参照
220     private void button2_Click(object sender, EventArgs e)
221     {
222         if (upc[1] == 0)
223         {
224             countud = count - 1;
225         }
226
227         num[1]++;
228         labelDisp(prod[countud], countud);
229         setValues(countud);
230         upc[1]++;
231
232     }
```

次に，フォームデザイナー上で button11 及び button12 をダブルクリックして，イベントハンドラ button11_Click 及び button12_Click の処理内容を次の様に記述する。

ここで，if 文に於ける条件の設定は，ダウンボタンが押された数が，アップボタンが押された数を上回らない，即ち，配列 num の要素の値が負にならない様に工夫している。

＊ 最初にアップボタンが押されると配列 upc の要素の値は 1，そこでダウンボタンを押された場合，配列 downc の要素の値は 0 であることに注意する。

```
        1 個の参照
234     private void button11_Click(object sender, EventArgs e)
235     {
236         if (upc[0] > downc[0])
237         {
238             num[0]--;
239             labelDisp(prod[countud], countud);
240             setValues(countud);
241             downc[0]++;
242         }
243     }
```

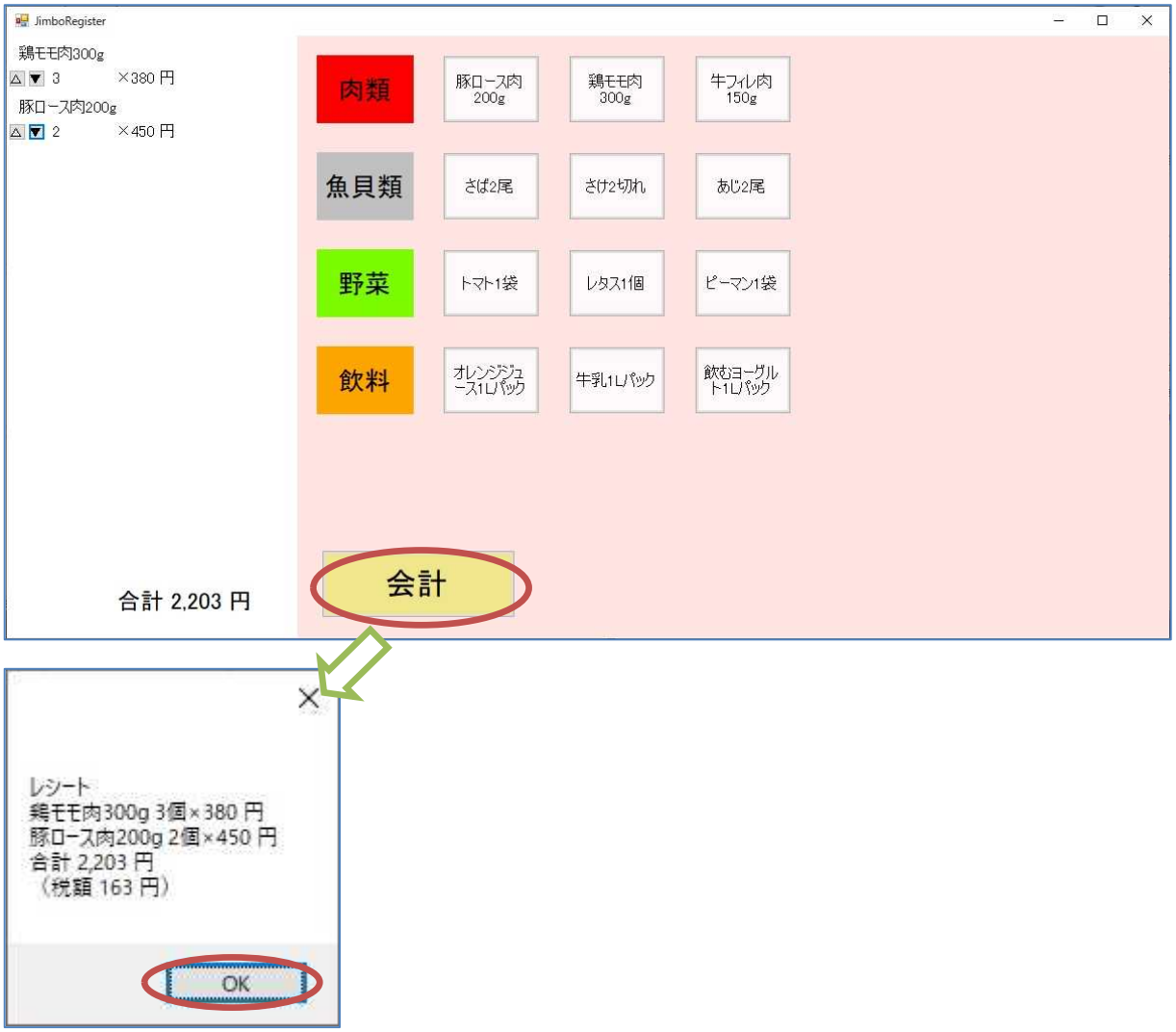
```
        1 個の参照
245     private void button12_Click(object sender, EventArgs e)
246     {
247         if (upc[1] > downc[1])
248         {
249             num[1]--;
250             labelDisp(prod[countud], countud);
251             setValues(countud);
252             downc[1]++;
253         }
254
255     }
```

2) 実行

ここで，『Form1.cs の保存』ボタンを押してから，『開始』ボタンを押して，プログラムを実行する。

先ず，1 件目の商品ボタンを押してからアップボタン及びダウンボタンの動作を確かめ，次に，2 件目の商品ボタンを押してからアップボタン及びダウンボタンの動作を確かめる。

最後に会計ボタンをクリックする。



提出物：

1）フォームのデザインファイル Form1.Designer.cs をメールに添付して提出する。

2）コードエディタで編集したソースファイル Form1.cs をメールに添付して提出する。

3）実行して，アプリが起動後，button22 をクリックした後，アップボタン及びダウンボタンの動作を確かめ，続いて button21 をクリックした後，アップボタン及びダウンボタンの動作を確かめて，両者とも商品件数を 2 以上とした状態のスクリーンショット第 8 回実行結果.jpg（.png も可）をメールに添付して提出する。

4）上の状態の後，button33 をクリックして表示されたレシートのスクリーンショット第 8 回レシート.jpg（.png も可）をメールに添付して提出する。

5）質問を記述したファイル Prog2_Questions_8th.txt に解答を書き込んで保存し，メールに添付して提出する。